Report Matthew Montoya

Running environment: Dr. Java
Computer: MacBook Pro (Retina, 13-inch, Early 2015)
Processor: 2.9 GHz Intel® Core™ i5 "Broadwell"
Memory: 8.00 GB
OS: macOS High Sierra (Version 10.13.2)
Java version: Version 8 Update 111

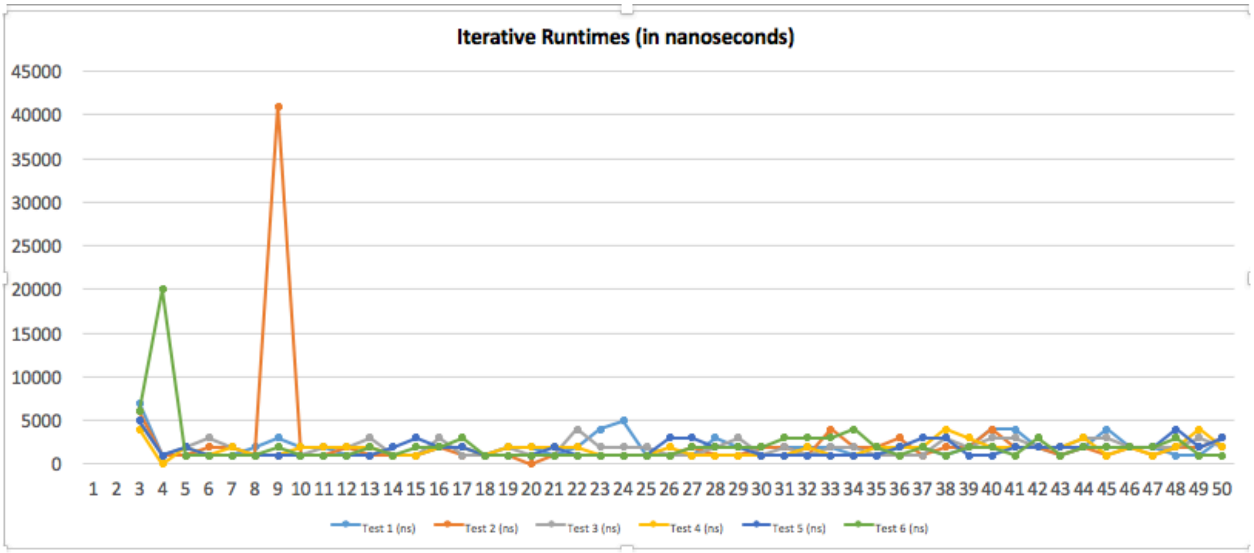
In order to generate the runtime plots for the Excel spreadsheet, it was necessary to first identify which data plots were needed. Using the information provided on the assignment sheet, the methods were developed by doing the following:

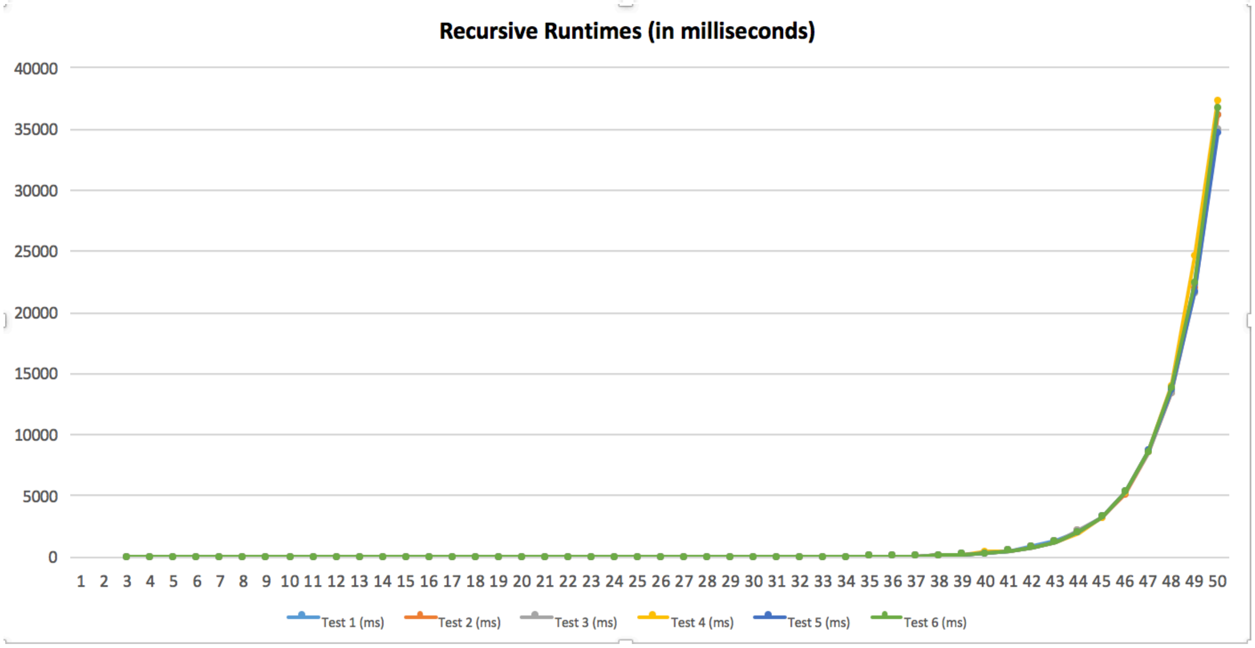
- Calculating the nth Fibonacci number iteratively and recursively
    1. Determine the range of numbers to be calculated
        - E.g. The 3rd – 50th numbers in the sequence
    2. Begin record the system time in both milliseconds and nanoseconds
    3. Call the method to determine the value of the *n*th Fibonacci number
        - Recall that if(*n*==2), it is really *n*[2]. Fibonacci sequence will start at position 0
    4. When the method returns the *n*th Fibonacci number, record the system time in milliseconds and nanoseconds
    5. Determine the elapsed time by subtracting the beginning timestamp from the ending timestamp
    6. Print *n*, the Fibonacci numbers (in ascending order), as well as the elapsed time it took for the system to calculate each respective number
        - This will identify when the machine has finished calculating all the specified numbers in the sequence

NOTE: The original methods were done in one program. However, after executing for the first time, there was a concern that running the program on a beta OS was too much for the machine to handle. The methods were then split into separate programs. After executing multiple times, it was discovered that the machine was handling the processes just fine and due to the nature of the recursive calls, the elapsed runtimes were *normal*. Laziness ensued, and the methods were kept in separate programs for the programmer's sake and sanity.

- <u>Calculating the average elapsed time</u>
    1. Copy the times into an Excel Spreadsheet
        - Use either milliseconds or nanoseconds, depending on the runtimes
    2. Run the program “a number of time” to average out the data
        - Average it out visually by plotting all data points in Excel and converting the data into a graph (copy and paste will work fine)
            - The x-axis will be the *n*th Fibonacci number to calculate
            - The y-axis will be the elapsed time

NOTE: To preserve scaling proportions, the graphs were screenshotted and pasted as JPEGs into this document. They can be viewed below.





The big-oh notation for *FiboIterative* is $O(n)$ and $O(2^n)$ for *FiboRecursive*. The number of instructions in each *FiboIterative* method call directly affect the speed of the program, which is executed *n* times. Looking at the graph for *FiboRecursive*, we can tell that it begins to grow exponentially (2^n), whereas *FiboIterative* stays generally linear, with some exceptions, detailed below.

Standard factors that can influence the accuracy of the results include background services, time of day (computer updates and background services), computer manufacturer, and overall system integration (such as designing the hardware and software to work in harmoniously with one another). Doing this activity on a computer with a pre-release operating system was not the best way to conduct an experiment, as background services may be utilizing more system services that would otherwise not be used on stable OS releases. While running the iterative tests, the machine had various programs actively running on the side. With macOS 10.12.2 beta 1, there is an issue with Facebook's Messenger client hanging for a while before quitting properly. Following the first iterative test, this program was attempting to quit and hung as expected. It is unknown if the time delay at the 9th Fibonacci is a direct result of this action.

Aside from the active background services, it should be noted that the laptop was plugged into an external monitor during the testing. While writing this report, the runtimes have since dropped off. It's possible this is related to the computer no longer needing to conduct graphics-intensive processes. It should also be noted there is a two-hour delay between the test time and the original writing of this report (5pm vs 7pm).